World Applied Sciences Journal 21 (Special Issue of Applied Math): 106-111, 2013
ISSN 1818-4952;

DOI: 10.5829/idosi.wasj.2013.21.am.21131

New Diffie Hellman Key Exchange Primitive Based Upon Binary Field

Uma S. Kanniah and Azman Samsudin

School of Computer Sciences, Universiti Sains Malaysia, 11800, Pulau Pinang, Malaysia

Abstract: One of the earliest public-key algorithm is Diffie Hellman key exchange algorithm. The Diffie Hellman key exchange allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. However, one drawback of this algorithm is the inherently expensive exponential calculation that provides the needed one-way trapdoor mechanism. Hence, an alternative approach is proposed whereby the key exchange will no longer depend on the prime field instead the newly proposed algorithm will now depend on the binary field as it is known that binary field operations are inexpensive and provides notably faster computations. The result evidently shows that the binary field Diffie Hellman performs almost twice as fast as the prime field Diffie Hellman. In general, this result serves as an example in which the binary field can benefit the mainstream cryptography.

Key words: Public Key Cryptography (PKC) . secure key exchange . finite field . binary field

INTRODUCTION

Public key cryptography (PKC) and related standards and techniques lies beneath security features of many application products related to communication. In PKC, each user that takes part in the communication will be given a pair of keys known as the public key and private key. The keys are bundled together with a set of operations associated with the keys to perform cryptographic operations. Only the particular user knows the private key whereas the public key is distributed to all users taking part in the communication.

The main PKC's cryptographic primitives are categorized into four major components. This includes key exchange, encryption, digital signature and zero knowledge. The details of each primitive are elaborated in Fig. 1. In this paper, we will be focusing on key exchange as this primitive is indeed a crucial element in PKC. Hence, the best known algorithm that supports this key exchange is the Diffie-Hellman key exchange which will be further discussed in the following section.

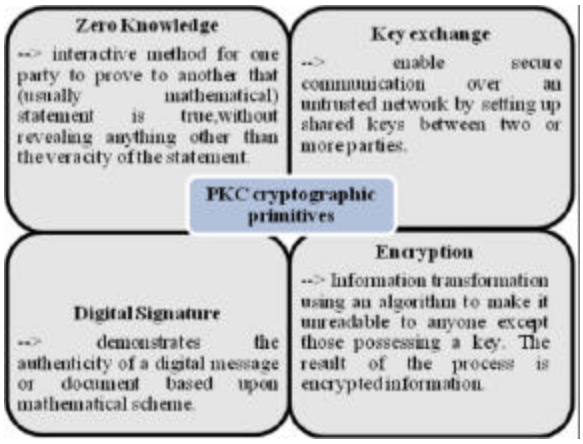

Fig. 1: Main public key cryptographic primitives

DIFFIE HELLMAN OVERVIEW

Whitfield Diffie and Martin Hellman discovered the Diffie-Hellman (DH) algorithm in 1976. This DH algorithm happens to be an amazing and ubiquitous algorithm that is found in many secure connectivity protocols on the Internet. In addition, DH is a method for securely exchanging a shared secret between two parties, in real-time as well as over an entrusted network. DH ensures that shared key plays an important role between two parties who may not have ever communicated previously thus providing surety that they can encrypt their communications. Evidently, this criteria has been by several protocols which includes Secure Shell (SS), Secure Socket Layer (SSL) as well as Internet Protocol Security (IPSec).

Mathematically, DH key exchange are based upon the modulus arithmetic precisely calculation over a prime field. DH key exchange algorithm starts with Alice and Bob's wish to share a secret key for use in a symmetric cipher yet their source of communication is insecure as each pieces of their exchangeable information are being viewed by another third person

Corresponding Author: Uma S. Kanniah, School of Computer Sciences, Universiti Sains Malaysia, 11800, Pulau Pinang, Malaysia

known to be Eve. The key in solving such a dilemma was to adapt the DH key exchange as the brilliant insight of DH relies on the difficulty of solving the discrete logarithm problem (DLP).

The first step in DH based key exchange is for Alice and Bob to agree on a large prime p and a nonzero integer g modulo p. Alice and Bob will ensure that the values of i and g to be publically known. The main concern here is to ensure that Alice and Bob chooses a relatively large g such that its order in $\mathbb{F}^*_p$ is a large prime. The following step is for Alice to pick a secret integer a and making sure that she does not reveal it to anyone whilst at the same time Bob will also pick an integer b where he keeps it secret. Thus Bob and Alice will use their secret integers to compute their public keys.

$$\underbrace{A = g^a \pmod p}_{\text{Alice computes this}} \quad \text{and} \quad \underbrace{B = g^b \pmod p}_{\text{Bob computes this}}$$

Thus, it proceeds by Alice sending A to Bob and Bob sends B to Alice. It's notable that Eve gets to see the values of A and B since the values are being sent through an insecure communication channel. Hence, Alice and Bob will once again use their private keys in computing the shared key.

$$\underbrace{A' = B^a \pmod p}_{\text{Alice computes this}} \quad \text{and} \quad \underbrace{B' = A^b \pmod p}_{\text{Bob computes this}}$$

The final value that the both parties compute will eventually be the same since, $A' = B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b = B' \pmod p$.

Although DH key exchange algorithm is secured due to the fact that its hard problem is related to the DLP problem, there are uncertainties when it comes to speed of the overall execution of the key exchange. This is due to the fact that as the number of bits for the keys increases the longer will the execution process takes place. Hence, in this particular research we will try to enhance the DH key exchange algorithm by proposing an alternative binary field compared to the currently prime field based.

RELATED WORK

Binary field: Finite fields in the order 2^m are known as binary fields. One way of constructing $F_2{}^m$ is to use a polynomial basis representation. Here, the elements of $F_2{}^m$ are the binary polynomials (polynomials whose coefficients are in the field $F_2 = \{0,1\}$) of degree at most m- based on [1]:

$$F_2{}^m = a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots + a_2 z^2 + a_1 z + a_0 : a_i \in \{0,1\} \quad (1)$$

An irreducible binary polynomial $f(z)$ of degree m is chosen such a polynomial exists for any m [1]. Irreducibility of $f(z)$ means that $f(z)$ cannot be factored as a product of binary polynomials each of degree less than m [1]. Addition of field elements is the usual addition of polynomials, with coefficient arithmetic performed modulo 2 whereby multiplication of field elements performs modulo e reduction polynomial f(z) [1]. Therefore, for any binary polynomial a(z), a(z) mod f(z) shall denote the unique remainder polynomial r(z) of degree less than m obtained upon long division of a(z) by f(z); this operation is called reduction modulo f(z) as stated in [1].

Primitive binary polynomials: A primitive polynomial of degree n over GF (2) is useful for generating a pseudo random sequence of n tuples of zeros and ones as mentioned in [2]. Hence, if the polynomial has a small number k of terms, then the sequence is easily computed yet for crypto logical applications it is often necessary to have the primitive polynomials with k larger than one can find in the existing tables based upon [2]. For example, Zierler and Brillhart in [3, 4] have calculated all irreducible trinomials of degree n = 1000, with the period for some for which the factorization of $2n$-1 is known.

According to Zivkovic in [5], the process begins by allowing F_q to denote the finite field of order $q = p^n$; whereby p is prime and n=1. The multiplicative group $F_q{}^*$ of nonzero elements of F_q is cyclic and a generator of $F_q{}^*$ is called a primitive element. A monic irreducible polynomial whose roots are primitive elements is called a primitive polynomial. As explained by Golomb in [6], it is known that the binary (over GF(2)) sequence $\{a_n\}_{n=0}$ satisfying the linear recurrent relation $a_{k+n} = \sum_{i=0}^{n-1} a_{k+i} f_i$ possesses good statistical properties if its characteristic polynomial $f = \sum_{i=0}^{n} f_i x^i$ is primitive. For example, the period length of such sequence (so called m-sequence) is $N = 2n$ -1, the difference between the number of ones and zeros across the period is exactly one and each n-tuple from $\{0,1\}^n$ except 0 appears exactly once in one period. The two sequences $\{a_{n+r}\}$ and $\{a_{n+s}\}$, $0 = r < s = N$, are mutually orthogonal, i.e., the equality

$$\sum_{n=0}^{N} -1^{a_{n+r}} -1^{a_{n+s}} = \begin{cases} N+1, \text{ for } 0 = r = s = N \\ -1, \text{ for } 0 = r < s = N \end{cases} \quad (2)$$

holds. The m-sequences are used for obtaining uniformly distributed random numbers as mentioned in [7]. Another field where m-sequences are widely used is cryptology [4]. Quality of m-sequences grows with n and therefore there is a need to obtain primitive polynomials of degree as large as possible.

Generation of primitive polynomials is performed by testing primitivity of the sequence of trial polynomials from the given set. Here we deal with the set of polynomials f of degree n with t terms, for given n and odd t, with the constraint $f(0) = 1$. The number t is usually small, to enable simple calculation of the corresponding linear recurrent sequence. The sequence of trial polynomials is formed using the linear recurrent sequence of order 127 as a source of random numbers. The primitivity test of a given polynomial f is effectively performed using the following set of conditions as shown in [4].

$$f(0) = f(1) = 1 \quad (3)$$

$$min\{k | f | x^{2^k} - x = n \quad (4)$$

$$for\ all\ prime\ p | 2n - 1\ f | x^{2^{n-1/p}} - 1 \quad (5)$$

Equation (3) eliminates polynomials divisible by x and $x + 1$. As t is odd for trial polynomials, this condition is automatically fulfilled. The polynomial $x^{2^k} - x$ is equal to the product of all irreducible polynomials of degree dividing k and therefore the irreducible polynomials satisfy (3) and (4). The inverse is not true: a polynomial equal to the product of different irreducible polynomials of degrees dividing n satisfy (3) and (4) (and it is not irreducible). The number of irreducible polynomials of degree n equals to

$$\frac{1}{n} \sum_{d|n} \mu(d) 2^{\frac{n}{d}} \quad (6)$$

where μ is the Moebius function, defined by

$$\mu(n) \begin{cases} 1\ if\ n = 1 \\ (-1)^r\ if\ n\ is\ the\ product\ r\ distinct\ primes \\ 0\ if\ n\ is\ divisible\ by\ the\ square\ of\ a\ prime \end{cases} \quad (7)$$

To check if f satisfies (4), it is necessary to calculate n residues $x^{2^k} - x \pmod f$. Squaring in GF (2) is simple, because $(a+b)^2 = a^2+b^2$. After each squaring, a residue is calculated from the division of a polynomial of degree at most $2n - 2$ by f. The fact that f is sparse is used to perform division more efficiently.

The total number of elementary (in GF(2)) operations needed to test (4) is bounded by $O(tn^2)$, which is not small, having in mind the number of polynomials that need to be checked. The problem is solved in the usual way whereby in (4) it is modified by previously checking the conditions

$$(f, x) = 1, 2 = k = 12 \quad (8)$$

$$(f; f') = 1 \quad (9)$$

where (f, g) is the greatest common divisor of f and g, which is computed by the Euclidean algorithm. This makes the complete test in (4) more complicated, but for the large part of trial polynomials roughly around 85% according to the estimation obtained from [3] it is ended by (8). Equation (9) eliminates the polynomials divisible by the square of a polynomial. The numerical complexity of finding the factorization of $2n$-1 is very large. This makes it unreasonable to include the factorization as a part of the primitivity check.

The efficiency of the primitivity check depends also on the order in which the prime factors of $2n$ -1 are used in (5) whereby the check is carried out for the small factors p first, because according to Lidl and Niederreiter as shown in [7] the probability that (5) is not satisfied is greater for small than for large prime factors.

FRAMEWORK

As it is known, existing DH key exchange algorithm relies on the prime field in key exchange therefore here we are proposing an alternative field that we can work with. Hence, binary field is chosen as binary field provides a better alternative to the existing method as generally, the binary field performs bit-shifting operations which are extremely fast.

The key idea here is literally to transform the DH key exchange algorithm from prime field to binary field whereby both this fields are incorporated in number theory. This mathematical approach indeed provides an alternative to the existing method in the representation of DH key exchange. Therefore, an alternative algorithm is proposed whereby the key exchange will no longer depend on the prime field instead the newly proposed algorithm will now depend on the binary field. This is due to fact that it is known that binary field operations are inexpensive and provides notably faster computations.

In addition, the existing number theory based DH key exchange will also be enhanced into binary based field providing a key element in accelerating the performance of DH key exchange. The execution time

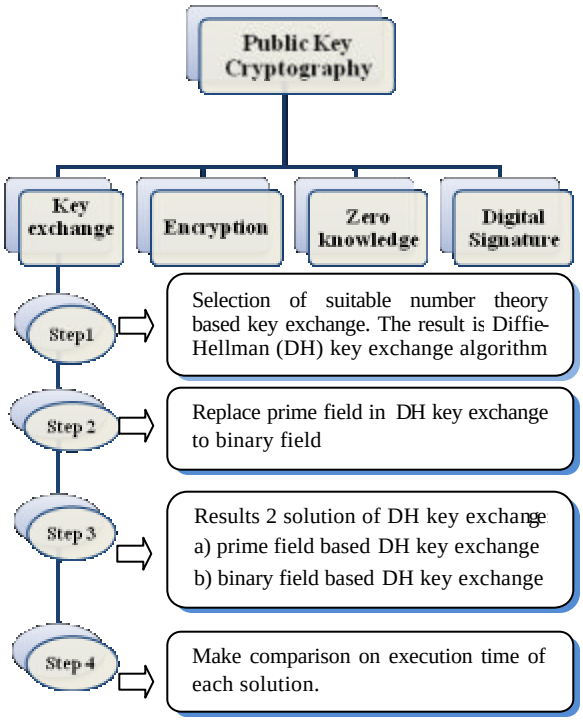

Fig. 2: Proposed framework

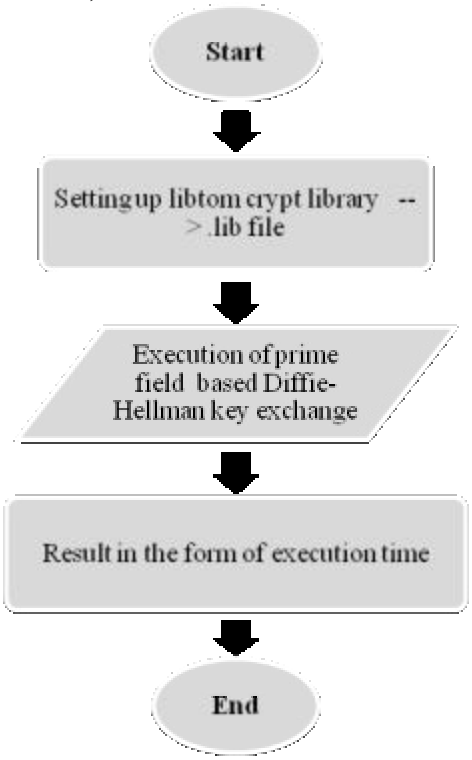

Fig. 3: Diffie-Hellman key exchange based on LibTom library

taken to perform prime field based DH key exchange and binary field based DH key exchange is accumulated and are later analyzed. The simplified version of the proposed framework is summarized in Fig. 2.

IMPLEMENTATION

The implementations are formed into 3 segments. The first segment will set up the existing number theory based library on the key exchange cryptographic primitive. The implementation process takes place by simply creating 2 different libraries whereby one denotes the prime field based library and the other denotes the prime polynomial based library. As we mentioned earlier, the prime field based library that will be incorporated in this research is LibTom library.

The reason behind choosing this library is due to the fact that the library code is written in simple C that are easily understandable and is user friendly. This prime field based library is also easy to work upon for modifications. In addition, the chosen LibTom library has also worked on cryptographic primitives mainly on key exchange. LibTom provides implementation codes on DH which are written in subproject known to be LibTomcrypt. The explanations on the usage of this library are shown in detail on Fig. 3.

The second segment of our implementation involves the proposal of a polynomial based library, more precisely a library designed in finding primitive polynomial. The generated primitive polynomials will later be transported into the third and the final segment.

The final segment of the implementation is relatively based upon the results obtained from the execution of the earlier segment. Thus, by finding the primitive polynomial, these polynomials will be used in the DH key exchange process. Alternatively, these primitive polynomials generated prime polynomials will be later be transformed into binary field prior to the key exchange process. This is mainly done in hopes that the binary field based key exchange will fasten the computation of the DH based key exchange. This is certain as we know, the bit-shifting operation that takes place in the binary field are extremely fast thus affecting the overall performance of the key exchange. In addition, this method will also minimize the operational cost of the entire key exchange process. The detailed implementation process of this method is explained in Fig. 4.

RESULTS AND TESTING

Here, we present the results obtained from the execution of the DH key exchange in completely two different perspectives. The key exchange is written using totally two different approaches. As mentioned

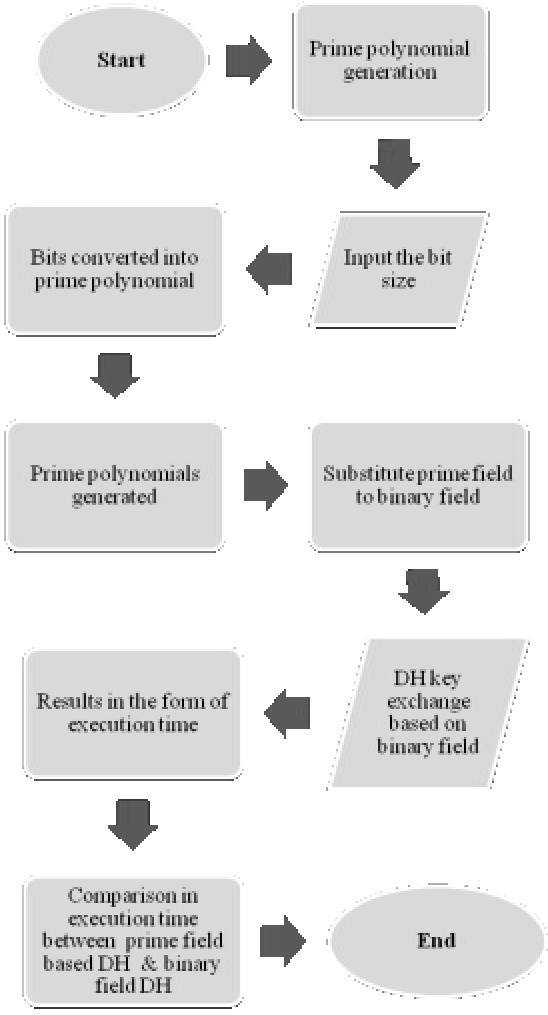

Fig. 4: Diffie-Hellman key exchange based upon binary field

Table 1: Computation time tested on prime generations in both number theory and binary polynomial

	Total timings (ms)	
Size (bits)	Number theory based prime generation	Primitive binary polynomial based prime generation
64	0.31	0.28
128	1.05	0.59
256	2.02	1.18
512	3.41	2.42
1024	4.23	3.91

ms = milliseconds

earlier, the first approach of the key exchange will be based upon prime field using an existing cryptographic library known as LibTom.

Table 2: Computation time tested on diffie-hellman key generation based upon number theory and primitive binary polynomial

	Total timings (ms)	
Size (bits)	Diffie-Hellman key generation based on number theory	Diffie-Hellman key generation based on binary primitive polynomial
64	0.31	0.19
128	1.18	0.62
256	24.77	10.89
512	32.04	20.23
1024	73.54	44.45

ms = milliseconds

Table 3: Computation time tested on diffie-hellman key exchange based upon number theory and primitive binary polynomial

	Total timings (ms)	
Size (bits)	Diffie-Hellman key exchange based on number theory	Diffie-Hellman key exchange based on binary primitive polynomial
64	0.34	0.20
128	1.20	0.58
256	28.18	11.09
512	32.88	20.56
1024	76.80	44.14

ms = milliseconds

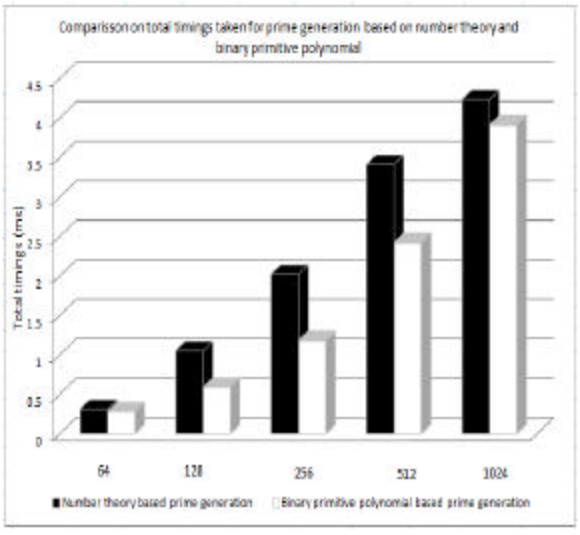

Fig. 5: Timings obtained for prime generation based upon number theory and primitive polynomial

The second approach which adapts the primitive binary polynomials will also be used on the testing of DH key exchange algorithm to calculate the execution

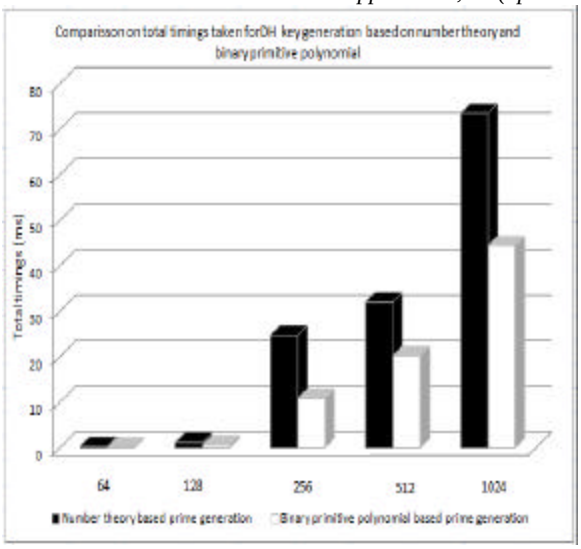

Fig. 6: Timings obtained for DH key generation based upon number theory and primitive polynomial

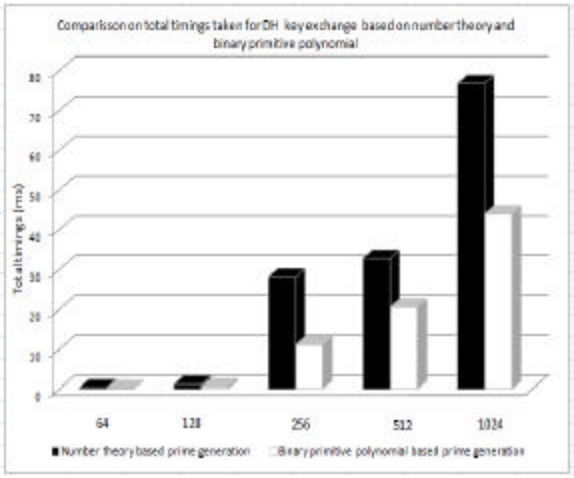

Fig. 7: Timings obtained for DH key exchange based upon number theory and primitive polynomial

time taken for this process. The results are tabulated in Table 3 whereby the execution times are calculated in milliseconds (ms). This program is developed in Microsoft Visual C++ and the specification of the machine used for the testing purposes is Intel Core, 2.9GHz Processor using Microsoft Windows XP Professional Edition.

The results are categorized into 3 parts. The first part shows the result of prime generation of DH based upon number theory and binary polynomial. The second part of the result will focus on the key generation and the final part of the result will focus on the key exchange of DH.

CONCLUSION

The inclusion of primitive polynomial as an alternative mathematical approach as certainly reduced the total computation whereby in primitive polynomial, the binary field played a vital role as it performs bit-shifting operations which are extremely fast. Generally, based upon the result it can be summarized that the existing prime field based DH key exchange has been enhanced into binary field. In addition, we have managed to speed up the execution process of DH key exchange. The overall performance indicates that there are improvements in total execution time in DH key exchange based upon primitive polynomial which are faster than the originally prime field based DH key exchange.

REFERENCES

1. Scott, M., 2007. Optimal Irreducible Polynomials for GF(2^m)Arithmetic. Cryptology ePrint Archive.
2. Tausworthe, R.C., 1965. Random Numbers Generated by Linear Recurrence Modulo-Two. Math. Comp., 19: 201-209.
3. Zivkovic, M., 1993. A Table of Primitive Binary Polynomials. Math. Comp., 92: 1368-1369.
4. Zivkovic, M., 1993. Table of Primitive Binary Polynomials. II, Math. Comp., 93: 368-372.
5. Zivkovic, M., 1995. Generation of Primitive Binary Polynomials. International Conference on Algebra, Logic and Discrete Mathematics.
6. Golomb, S.W., 1967. Shift Register Sequences, Holden Day, San Francisco.
7. Lidl, R. and H. Niederreiter, 1983. Finite Fields, Encyclopedia Math. Appl., Addison-Wesley, Reading, Mass., Vol: 20.